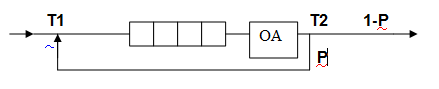
# Лабораторная работа №4. Обработка очередей

**Условие задачи**

Система массового обслуживания состоит из обслуживающего аппарата (ОА) и очереди заявок.



Заявки поступают в "хвост" очереди по случайному закону с интервалом времени Т1, равномерно распределенным от 0 до 6 единиц времени (е.в.). В ОА они поступают из "головы" очереди по одной и обслуживаются также равновероятно за время Т2 от 0 до 1 е.в., Каждая заявка после ОА с вероятностью Р=0.8 вновь поступает в "хвост" очереди, совершая новый цикл обслуживания, а с вероятностью 1-Р покидает систему. (Все времена – вещественного типа). В начале процесса в системе заявок нет.

**Техническое задание**

Смоделировать процесс обслуживания до ухода из системы первых 1000 заявок. Выдавать после обслуживания каждых 100 заявок информацию о текущей и средней длине очереди. В конце процесса выдать общее время моделирования и количество вошедших в систему и вышедших из нее заявок, среднее время пребывания заявки в очереди, время простоя аппарата, количество срабатываний ОА. Обеспечить по требованию пользователя выдачу на экран адресов элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

**Исходные данные**

Структура данных для хранения очереди. Время поступления заявок в очередь, время обработки заявок – вещественного типа.

**Выходные данные**

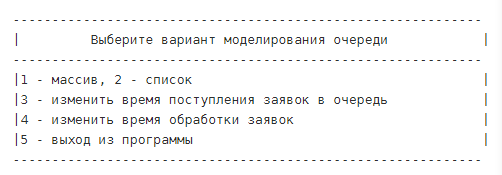
После обслуживания каждых 100 заявок выводится информация о текущей и средней длине каждой очереди, а в конце процесса – общее время моделирования, количество вошедших в систему и вышедших из нее заявое, среднее время пребывания заявки в очереди, время простоя аппарата, количество срабатывания аппарата, процент погрешности работы(по входу и выходу), при реализации списком – адреса свободных областей памяти.

**Возможные ошибки пользователя**

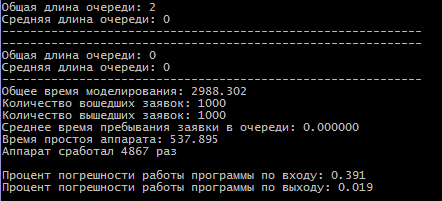
|  |  |  |
| --- | --- | --- |
|  | Тип ошибки | Сообщение |
| 1 | Неверный выбор пункта меню | Такой команды нет |
| 2 | Неверный тип введенных данных | Неверный тип введенных данных |
| 3 | Переполнение | Переполнение очереди |

**Интерфейс программы**

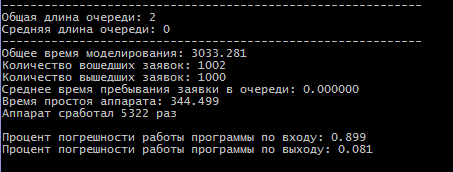
Меню



Моделирование на массиве



Моделирование на списке



**Структуры данных**

*Очередь на массиве*

typedef struct Array

{

double storage[CAPACITY];

int pin;

int pout;

int size;

}array;

*Очередь на списке*

struct node

{

double value;

struct node \*next;

};

struct lqueue

{

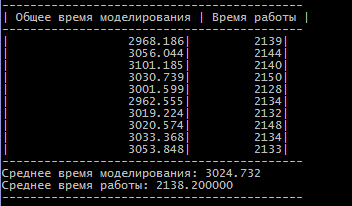
struct node \*pin, \*pout;

int size;

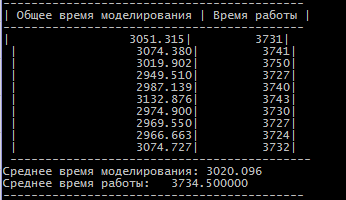
};

**Сравнение эффективности**

*Для массива*



*Для списка*



При реализации на массиве выигрыш 1,5-2 раза. При работе программы возникает фрагментация. Одной из причин возникновения является разница во времени обработки и времени добавления(время обработки горздо меньше времени добавления элемента в очередь).

**Контрольные вопросы**

**Что такое очередь?**

Очередь – это последовательный список переменной длины, включение в который идет с одной стороны (с хвоста), а исключение – с другой стороны (с головы).

**Каким образом, и какой объем памяти выделяется под хранение очереди при различной ее реализации?**

Массив: Выделяется последовательная область памяти константного размера. Выделение памяти происходит в начале работы программы. При необходимости память перевыделяется.

Список: Память под элемент очереди выделяется непосредственно в процессе его добавления. Объем памяти, который занимает очередь, изменяется в процессе выполнения программы и напрямую зависит от количества элементов в очереди в каждый момент времени.

**Каким образом освобождается память при удалении элемента из очереди при ее различной реализации?**

Массив: освобождение памяти происходит в конце работы программы (или при удалении очереди). При удалении эл-та из очереди происходит только смещение указателя.

Список: При удалении элемента из очереди происходит освобождение памяти, которая была выделена под этот элемент.

**Что происходит с элементами очереди при ее просмотре?**

При просмотре очереди хвостовой элемент из нее удаляется.

**Каким образом эффективнее реализовать очередь? От чего это зависит?**

Если необходимо избежать фрагментации памяти, то следует использовать очередь на массиве. Также выбор способа реализации зависит от того, известно ли заранее количество эл-тов в очереди и насколько оно велико.

**Каковы достоинства и недостатки различных реализаций очереди в зависимости от выполняемых над ней операций?**

При реализации очереди массивом не возникает фрагментации памяти, однако может произойти переполнение очереди, а также затрачивается дополнительное время на сдвиг элементов. При реализации очереди списка затрачивается большее кол-во времени при добавлении нового элемента.

**Что такое фрагментация памяти?**

При последовательных запросах на выделение и освобождение памяти под элемент не всегда выделяется память, которая была только что освобождена.

**На что необходимо обратить внимание при тестировании программы?**

При тестировании программы необходимо обратить внимание на переполнение очереди, фрагментацию памяти при реализации очереди списком.

**Каким образом физически выделяется и освобождается память при динамических запросах?**

Программа дает запрос ОС на выделение блока памяти необходимого размера. ОС находит подходящий блок, записывает его адрес и размер в таблицу адресов, а затем возвращает данный адрес в программу. При запросе на освобождение указанного блока программы, ОС убирает его из таблицы адресов, однако указатель на этот блок может остаться в программе. Попытка считать данные из этого блока моет привести к непредвиденным последствиям, поскольку они могут быть изменены.